L1 Calo DAQ Requirements

Murrough Landon – 17 Sep 1999

- Our Timescales
- Hardware Summary
- Prototype Systems
- Final System
- Calibration
- Software Options
- Miscellaneous Comments
- Summary of Requirements

Our Timescales

- Various prototype modules over next 18 months
- This month: Data Sink & Source (DSS) module
- End of 1999: prototype ROD to be tested with DSS
- End of 2000: trigger processor modules
- 1999-2001: subsets of complete slice tests

Hardware Summary

CPUs with VME access

- Currently 68K and PPC based LynxOS systems
- Short term: Linux PCs using LynxOS as VME bridge
- New hardware? Need to know it will be supported by DAQ group up to 2004
- Would like same/similar setup in each of six institutes

Present Test Setups

- Several VME crates (6U, 9U, various backplanes)
- Linked by VICbus (or similar)
- Controlled by a single CPU (ie could be single LDAQ)

Prototype Tests

Early Prototype Tests

- Single CPU system running all DAQ processes
- Want to read out RODs **and** other modules
- Typical operation mode: feed in known inputs, compare with expected outputs, generate “trigger” on mismatch
- (Hopefully!) low or no data acquisition rate
- Would like to start developments in a framework which would evolve to the final DAQ system
- Or at least based on common design

Prototype L1 Calo Slice

- Multiple crates controlled by separate CPUs
- Need to read synchronised events from multiple ROD/processor crates
- Prototype modules (ROD, DSS) will have facilities to store selected events for subsequent readout
- Aim to have prototype of final software
- Prototype calibration and setup procedures
- Complete slice of Level 1 Calo Trigger
- Complete slice tests with ROIbuilder, Level 2, CTP, DAQ

Final System

RODs

- Two different types of ROD module
- PreProcessor RODs scattered among PP crates. Maybe NOT VME64ext?
- Trigger processor RODs in separate 6U(?) VME64ext crates

Final System

- Multiple trigger and ROD crates, one CPU per crate (probably)
- May still want to read non-RODs (calibration, testing)
- Many functions move from ROD crate to ROC, EF
- Some (single and multicrate) functions still done at ROD crate level?

Calibrations

Types of Calibration

- Typical procedure: run, stop, analyse, change parameter, iterate
- Standalone setup procedures (internal timing, cabling)
- Input signal cabling, timing, BCID parameters
- Energy calibration (pulser, particles)
- Trigger processor performance

Requirements

- Use one or more LAr and/or TileCal partition
- Control LAr/TileCal calibration systems (maybe both)
- BUSY network, TTCvi, local trigger, etc
- In some cases, readout of single ROD/processor crate is enough
- In many cases, readout of multiple crates is required

Software Options

Complete DAQ -1 Solution

- Pick and choose (most) backend core components
- Use single CPU Dataflow skeleton
- Use the hooks for our own readout and monitoring
- Short term requirement is more for functionality rather than performance

Do It (Partly) Ourselves

- If no suitable ROD crate DAQ, then...
- Reuse our existing buffer manager software
- New control and readout processes (using Backend components)
- New monitoring task(s)
- If possible follow DAQ -1 designs

Miscellany

Miscellaneous

- Subdetector run parameters in Integrated GUI: plugins?
- Saving dynamic changes to static configuration data?
- Updating database on local processors? Objectivity to scattered OKS files?
- Distribution of calibration parameters: IS?
- Subdetector monitoring: publish/subscribe to large chunks of data eg histograms (ROOT?) but also other data structures

Summary: Requests to DAQ group

Required DAQ Functionality

NOW Core (and maybe a few other) Backend DAQ components

NOW Framework for reading events from RODs and other modules (in a "single" crate)

NOW Framework for monitoring events

By 2001 Requirement for reading synchronised events from different crates (given mechanisms in the modules to support this)

- Short term requirement is more for functionality rather than performance
- DAQ group to support evolution from DAQ -1 to final DAQ system (eg recent script to aid migration to new backend APIs)
- Agreed set of hardware platforms and operating systems to be supported from now until commissioning of the final system.